Complexity of Manipulation in Premise-Based Judgment Aggregation with Simple Formulas

Robert Bredereck
TU Berlin
Berlin, Germany
robert.bredereck@tu-berlin.de

Junjie Luo
University of Chinese Academy of Science
Beijing, China
luojunjie@amss.ac.cn

ABSTRACT

Judgment aggregation is a framework to aggregate individual opinions on multiple, logically connected issues into a collective outcome. It is open to manipulative attacks such as Manipulation where judges cast their judgments strategically. Previous works have shown that most computational problems corresponding to these manipulative attacks are NP-hard. This desired computational barrier, however, often relies on formulas that are either of unbounded size or of complex structure.

We revisit the computational complexity for a large class of Manipulation problems in judgment aggregation, now focusing on simple and realistic formulas. We restrict all formulas to be clauses that are (positive) monotone, Horn-clauses, or have bounded length. For basic variants of Manipulation, we show that these restrictions make several variants, which were in general known to be NP-hard, polynomial-time solvable. Moreover, we provide a P vs. NP dichotomy for a large class of clause restrictions (generalizing monotone and Horn clauses) by showing a close relationship between variants of Manipulation and variants of Satisfiability. For Hamming distance based Manipulation, we show that NP-hardness even holds for positive monotone clauses of length three, but the problem becomes polynomial-time solvable for positive monotone clauses of length two.

ACM Reference Format:
Robert Bredereck and Junjie Luo. 2019. Complexity of Manipulation in Premise-Based Judgment Aggregation with Simple Formulas. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Alice is the head of a committee deciding on financial support for new startup companies. For her decisions, she uses publicly available evaluations of experts (judges) with respect to a set of basic features such as cult potential (c), marketability (m), high profitability (h), and strong competitors' existence (s). As a brilliant mathematician and economist, Alice developed a model that can reliably predict the success of the startup by putting the features into logical relation. For example, she defined two further composed features "market entering potential" as $e := \neg s \vee c$ and "short-term risk" as $r := \neg m \vee \neg h$. Her first idea was to make her decisions based on the majority on each feature, but she recognizes that she may obtain the following evaluations from three experts:

expert 1:	s	$\wedge$	$\neg c$	$\wedge$	m	$\wedge$	h	$\wedge$	$\neg e$	$\wedge$	$\neg r$
expert 2:	s	$\wedge$	c	$\wedge$	$\neg m$	$\wedge$	h	$\wedge$	e	$\wedge$	r
expert 3:	$\neg s$	$\wedge$	$\neg c$	$\wedge$	m	$\wedge$	$\neg h$	$\wedge$	e	$\wedge$	r

where the majority opinions are: s, $\neg c$, m, and h, but also $e = \neg s \vee c$ and $r = \neg m \vee \neg h$; an obviously paradox situation. Alice does a quick literature review and identifies her aggregation problem as "judgment aggregation" and the observed paradox as a variant of the well-known doctrinal paradox [16]. To avoid this paradox and since the experts are anyway better in evaluating basic features than evaluating composed features, she decides to adapt the concept of premise-based judgment aggregation rules [9] for her decision process: Basic features form the *premises*, composed features are *conclusions* (which logically connect premises). The aggregation process is performed only on the premises and conclusions are deduced from them. That is, the outcome in the above example is s, $\neg c$, m, h, and, hence, also $\neg e$ and $\neg r$.

Alice is happy with the aggregation process, but she is worried about the reliability of the results. For example, what if an expert made a mistake? Can she compute efficiently whether a set of important features remains stable even if some expert provided a wrong evaluation? What if an expert evaluated strategically or untruthfully due to bribery or lobbyism? Is it difficult for an expert to compute a successful strategy? Since Alice has only little knowledge about computer science, she consults Bob, her favorite algorithms and complexity expert. Bob does a quick literature review and identifies all questions posed by Alice as variants of Manipulation, which are computationally intractable.

Although the intractability of strategic evaluation and bribery seems to be good news, Alice is skeptical about the relevance of these results for her application. In her model, all formulas are length-two *Horn clauses*. All intractability results found by Bob, however, use rather complex or long formulas as conclusions, and, hence, do not apply in her situation. So Alice asks Bob to revisit the respective computational complexity results with respect to simple formulas as they occur in her model.

In this paper, we take up Bob's task and provide a fine-grained computational complexity analysis of Manipulation for judgement aggregation with simple formulas.

Related work. We refer to recent surveys [5, 11, 15, 17, 18] for a detailed overview on judgment aggregation. Dietrich and List [10] introduced strategic behavior to judgment aggregation. Following Bartholdi III et al. [1, 2], intractability of manipulative attacks is usually seen as "barrier against manipulation" and, hence, a desired property. Endriss et al. [12] were the first who analyzed the computational complexity of strategic behavior in judgment aggregation and showed that it is NP-hard for a judge to decide whether she can

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

cast a judgment set that influences the collective outcome in a beneficial way (assuming Hamming distance based preferences over judgment sets), even for the simple premise-based majority rule. Baumeister et al. [3, 4] continued this line of research and extended the results to the more general uniform premise-base judgment aggregation rules, and also initiated the analysis of further variants of strategic behavior for judgment aggregation, including further variants of Manipulation or cases where an external agent influences the structure (Control) or the opinions of the judges (Bribery), showing NP-hardness for most considered problems. For more details, we refer to a recent survey on strategic behavior in judgment aggregation [6]. Our work also fits well into the line of research initiated by the seminal paper of Faliszewski et al. [13] showing that the barrier against manipulative attacks sometimes disappears in context of restricted domains. In context of voting one usually considers restricted preference domains whereas we focus on restricted formulas.

Organization and Contributions. We analyze the computational complexity of variants of Manipulation in premise-based judgment aggregation with simple formulas. In particular, we consider Horn clauses (implication-like conclusions which for instance are fundamental in logic programming [8, 19]), (positive) monotone clauses, and clauses of bounded length. In Section 2 we describe the formal model and introduce our notation. In Section 3 we revisit the computational complexity for basic variants of Manipulation, showing that the restriction to clauses makes several variants, which were in general known to be NP-hard [4], polynomial-time solvable. Our first main result is a P vs. NP dichotomy for a large class of clause restrictions (generalizing monotone and Horn clauses) by showing a close relationship between variants of Manipulation and variants of Satisfiability. For details, we refer to Table 5 in our conclusion (Section 5). We revisit Hamming distance based Manipulation in Section 4. Our second main result is that for positive monotone clauses the problem becomes polynomial-time solvable for clauses of length $\ell = 2$ but remains NP-hard when $\ell = 3$. This is particularly surprising since Satisfiability is trivial for positive monotone clauses even for unbounded length. The latter result is reached by showing NP-hardness of a natural variant of Vertex Cover which we believe to be interesting on its own. The NP-hardness also holds for monotone or Horn clauses of length $\ell = 2$. Due to the lack of space, many proofs (of results marked with ($\star$)) are deferred to the full version of this paper.

2 MODEL AND PRELIMINARIES

We adopt the judgment aggregation framework described by Baumeister et al. [4] and Endriss et al. [12] and slightly simplify it for premise-based rules.

Premise-Based Judgment Aggregation. The topics to be evaluated are collected in the *agenda* $\Phi = \Phi_p \uplus \Phi_c$ that consists of a finite set of premises Φ_p (propositional variables) as well as a finite set of conclusions Φ_c (propositional formulas built from the premises using standard logical connectivities $\neg$, $\vee$, and $\wedge$).[1] The agenda does not contain any doubly negated formulas and is closed under complementation, that is, $\neg\alpha \in \Phi$ if and only if $\alpha \in \Phi$. An evaluation on the agenda is expressed as a *judgment set* $J \subseteq \Phi$. A judgment set is *complete* if each premise and conclusion is contained either in the negated or non-negated form and *consistent* if there is an assignment that satisfies all formulas simultaneously. The set of all complete and consistent subsets of Φ is denoted by $\mathcal{J}(\Phi)$.

Let $N = \{1, ..., n\}$ be a set of $n > 1$ *judges*. A profile is a vector of judgment sets $\boldsymbol{J} = (J_1, \ldots, J_n) \in \mathcal{J}(\Phi)^n$. We denote by $(\boldsymbol{J}_{-i}, J_i')$ the profile that is like $\boldsymbol{J}$, except that J_i has been replaced by J_i'. A *judgment aggregation procedure* for agenda Φ and judges $N = \{1, ..., n\}$ is a function $F : \mathcal{J}(\Phi)^n \to 2^\Phi$ that maps a profile $\boldsymbol{J}$ to a single judgment set, which is called a *collective judgment set*.

The probably most natural procedure is the majority rule, where an element in the agenda is contained in the collective judgment set if and only if it is contained in more than half of judgment sets in profile $\boldsymbol{J}$. Dietrich and List [9] introduced the *quota rule* as a generalization of the majority rule.

Definition 2.1 (Uniform Premise-based Quota Rule for $q \in [0, 1)$). A uniform premise-based quota rule $\mathrm{UPQR}_q : \mathcal{J}(\Phi)^n \to 2^\Phi$ divides the premises Φ_p into two disjoint subsets Φ_q and $\Phi_{\bar{q}}$, each containing every premise either in the negated or non-negated form. For each $\boldsymbol{J} \in \mathcal{J}(\Phi)^n$ the outcome $\mathrm{UPQR}_q(\boldsymbol{J})$ is the collective judgment set that contains every premise from Φ_q that appears at least qn times in the profile $\boldsymbol{J}$, every premise from $\Phi_{\bar{q}}$ that appears more than $n - qn$ times in the profile, as well as all conclusions that are satisfied by these premises.

In order to analyze the influence of judges on the outcome, we call a variable x *decided* by judge i if the judge can change the outcome with respect to x by changing J_i, that is, $x \in \mathrm{UPQR}_q(\boldsymbol{J}) \cap J_i$ and $\neg x \in \mathrm{UPQR}_q(\boldsymbol{J}_{-i}, ((J_i \setminus \{x\}) \cup \{\neg x\}))$ or $\neg x \in \mathrm{UPQR}_q(\boldsymbol{J}) \cap J_i$ and $x \in \mathrm{UPQR}_q(\boldsymbol{J}_{-i}, ((J_i \setminus \{\neg x\}) \cup \{x\}))$.
The following example formally restates the introductory example.

Example. The premise set Φ_p contains two parts $\Phi_q = \{s, c, m, h\}$ and $\Phi_{\bar{q}} = \{\neg s, \neg c, \neg m, \neg h\}$. The conclusion set Φ_c contains $\neg s \vee c$, $\neg m \vee \neg h$ and their negations. The profile is given as follows:

Judgment Set	s	c	m	h		$\neg s \vee c$	$\neg m \vee \neg h$
J_1	1	0	1	1		0	0
J_2	1	1	0	1		1	1
J_3	0	0	1	0		1	1
$UPQR_{1/2}$	1	0	1	1	$\Rightarrow$	0	0

In the table we use 1 or 0 to represent whether the formula is in the judgment set or not. As an example, $J_1 = \{s, \neg c, m, h, \neg(s \vee c), \neg(\neg m \vee \neg h)\}$. Since in this example $q = 1/2$, we first have that s, $\neg c$, m and h are included in the outcome $\mathrm{UPQR}_{1/2}(J_1, J_2, J_3)$. Or we just say $s = 1$, $c = 0$, $m = 1$ and $h = 1$. Then we get that $\neg s \vee c = 0$ and $\neg m \vee \neg h = 0$, which means $\neg(s \vee c)$ and $\neg(\neg m \vee \neg h)$ are included in the outcome. In this example variables c and m are decided by the third judge but s and h are not decided by the third judge.

Clause restrictions. We restrict the conclusions to be clauses (defined as disjunctions of literals). We define classes of clause restrictions based on a classification with respect to the number of positive and negative literals in a clause and consider a restricted variant of Satisfiability.

[1] This implies that the agenda is closed under propositional variables, that is, if ϕ is a formula in the agenda, then so is every propositional variable occurring within ϕ.

Definition 2.2. A clause set C is called a **standard-form clause set** if C is a union of some S_i^j, where S_i^j is the set of clauses which contain exactly i literals and exactly j of them are negative. Denote S_k^0 and S_k^k as M_k^+ and M_k^-.

Definition 2.3. Given a set C of clauses, C-Sat is the problem of deciding whether a given formula $C_1 \wedge \ldots \wedge C_m$ with $C_i \in C$ is satisfiable or not.

This classification is useful as most clause classes we care about can be defined as the union of some S_i^j. For example, positive monotone clauses can be denoted by $\bigcup_i S_i^0 = \bigcup_i M_i^+$, Horn clauses can be denoted by $\bigcup_{j \geq i-1} S_i^j = \bigcup_i (M_i^- \cup S_i^{i-1})$, and 3-CNF clauses can be denoted by $\bigcup S_3^j$. 3-Sat corresponds to $\bigcup S_3^j$-Sat.

3 BASIC MANIPULATION PROBLEMS

In this section, we analyze the computational complexity of problems modeling simple variants of strategic behavior of some judge. The core idea is that a judge might cast an untruthful judgment set in order to influence the collective judgment set towards some desired judgment set. Note that we provide alternative, simpler (yet equivalent) problem definitions compared to those known from the literature [4].[2] In contrast to Baumeister et al. [4] who focus on the assumption on the preferences of the manipulator over all possible outcomes, which requires rather technical concepts of preference relations between judgment sets, we take a different approach and directly model the requirements on the preferred outcome. For example, the simplest variant of manipulation from Baumeister et al. [4], UPQR-U-Possible-Manipulation, actually models the question whether the collective outcome is "robust against one judge providing a faulty judgment set" as asked by Alice in the introduction. Formally, we consider the following problems.

UPQR Manipulation basic variants (Problem names from[4] listed below.)
Input: An agenda Φ, a profile $\boldsymbol{J} = (J_1, \ldots, J_n) \in \mathcal{J}(\Phi)^n$, the consistent (possibly incomplete) desired set $J \subseteq J_n$, and a rational threshold $q \in [0, 1)$.

UPQR Robustness Manipulation (=UPQR-U-Possible-Manipulation[4])
Question: $\exists J^* : \mathrm{UPQR}_q(\boldsymbol{J}) \cap J \neq \mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*) \cap J$?

UPQR Possible Manipulation (=UPQR-CR-Possible-Manipulation[4])
Question: $\exists J^* : (\mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*) \cap J) \setminus (\mathrm{UPQR}_q(\boldsymbol{J}) \cap J) \neq \emptyset$?

UPQR Necessary Manipulation (=UPQR-CR-Necessary-Manipulation[4])
Question: $\exists J^* : \mathrm{UPQR}_q(\boldsymbol{J}) \cap J \subsetneq \mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*) \cap J$?

UPQR Exact Manipulation (=UPQR-TR-Necessary-Manipulation[4])
Question: $\exists J^* : J \subseteq \mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*)$?

Intuitively, the manipulator only cares about the formulas in the desired set J. UPQR Robustness Manipulation asks whether the manipulator can achieve a different outcome with respect to J. UPQR Possible Manipulation asks whether the manipulator can achieve an outcome that contains a formula from J which is not contained in the truthful outcome. UPQR Necessary Manipulation asks whether the manipulator can achieve an outcome that contains a formula from J which is not contained in the truthful outcome, and meanwhile contains all formulas that are in both J and the truthful outcome. UPQR Exact Manipulation asks whether the manipulator can achieve an outcome that contains all formulas from J.

Baumeister et al. [4] showed that all the four variants of UPQR Manipulation with the desired set being incomplete are NP-complete. However, a complex formula in conjunctive normal from is needed in the conclusion set in these reductions. In the following, we give a more refined analysis by considering how restricting the conclusions to different standard-form clause sets influences the computational complexity of UPQR Manipulation for all four basic variants.

3.1 Tractable Cases of Manipulation

We start our analysis with UPQR Robustness Manipulation and UPQR Possible Manipulation which turn out to be linear-time solvable when the conclusions are just simple clauses.

Lemma 3.1 (⋆). *UPQR Robustness Manipulation and UPQR Possible Manipulation with conclusions being clauses are solvable in linear time.*

Corollary 3.2. *UPQR Robustness Manipulation and UPQR Possible Manipulation with conclusions chosen from monotone clauses or Horn clauses are solvable in linear time.*

Next, we show that UPQR Necessary Manipulation boils down to solving a related Satisfiability problem.

Lemma 3.3. *UPQR Necessary Manipulation and UPQR Exact Manipulation with conclusions from clause set C can be solved by solving at most $|\Phi_C|$ instances of C-Sat.*

Proof. We show the result for UPQR Necessary Manipulation. The result for UPQR Exact Manipulation can be proven similarly. In UPQR Necessary Manipulation we should find a manipulated judgment set J^* such that $\mathrm{UPQR}_q(\boldsymbol{J}) \cap J \subsetneq \mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*) \cap J$. That is, the manipulated result $\mathrm{UPQR}_q(\boldsymbol{J_{-n}}, J^*)$ should not only contain one more target conclusion $C^* \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$, but also contain all formulas in $Q_0 = \mathrm{UPQR}_q(\boldsymbol{J}) \cap J$. So the problem is to check whether there exists a conclusion $C^* \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ such that a set $Q = Q_0 \cup \{C^*\}$ of conclusions can be satisfied by just controlling the values of variables which are decided by the manipulator.

We can simply try all possible $C^* \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ and for each C^* check whether all conclusions in $Q = Q_0 \cup \{C^*\}$ can be satisfied as follows. Every conclusion in Q is either a clause from C or a negation of a clause from C. To satisfy a negative clause (conjunction of literals) the values of all variables in this clause are fixed. For all negative clauses in Q, we first check whether they are consistent. This can be done in linear time. If all negative clauses in Q are consistent, then we get the value for all variables in them. Then, we need to check for every such variable whether the value is either the original value before the manipulation or the variable is decided by the manipulator. Otherwise, these negative clauses can not be satisfied. After this, we only need to check whether the remaining positive clauses in Q can be satisfied. Since all clauses in Q are chosen from C, the remaining problem forms an instance of C-Sat. □

[2] Our problem definitions slightly differ from those in the literature as we put the threshold value q as part of the input. For our polynomial-time algorithms, the quota is only interesting for computing which premises can be decided by the manipulator. Thus, it has no influence on the computational complexity. Our hardness reductions usually assume that $q = 1/2$, but they can all be adapted to work for any rational quota q.

Table 1: Instance of UPQR Necessary Manipulation with conclusion set $C = M^+_{k_1+1} \cup M^-_{k_2}$ for the proof of Lemma 3.7.

Judgment Set	x_1	...	x_n	y_1	y_2	...	y_{k_2}		$C_i^+ \vee y_1$	C_i^-	$\neg y_1 \vee \ldots \vee \neg y_{k_2}$
J_1	1	...	1	1	1	...	1		1	0	0
J_2	0	...	0	0	1	...	1		0	1	1
J_3	0	...	0	1	0	...	0		1	1	1
$UPQR_{1/2}$	0	...	0	1	1	...	1	$\Rightarrow$	1	1	0

Corollary 3.4. *If C-Sat is in P, then UPQR Necessary Manipulation and UPQR Exact Manipulation with conclusions from C are in P.*

From Corollary 3.4, we know that when C-Sat is in P, the corresponding problem UPQR Necessary Manipulation with conclusions chosen from C is also in P. In the next section we show that these two problems are actually polynomial-time equivalent for many clause classes.

3.2 Intractable Cases of Manipulation: Manipulation vs. Satisfiability

In this section, we give a full characterization for the computational complexity of UPQR Necessary Manipulation by showing that C-Sat and UPQR Necessary Manipulation with conclusions chosen from C are actually equivalent under polynomial-time Turing reductions when C is a standard-form clause set (see Definition 2.2 for the definition of standard-form clause).

Theorem 3.5. *For any standard-form clause set C, UPQR Necessary Manipulation with conclusions from C and C-Sat are equivalent under polynomial-time Turing reductions.*

In order to prove Theorem 3.5, we first consider for what kind of standard-form clause sets C, C-SAT is NP-complete.

Lemma 3.6. *($\star$) For a standard-form clause set C, C-SAT is NP-complete if and only if*

(1) there is a pair of i, j with $i \geq 3$ and $0 < j < i$ such that $M_2^+ \cup M_2^- \cup S_i^j \subseteq C$, or

(2) there is a pair of k_1, k_2 with $\max\{k_1, k_2\} \geq 3$ and $\min\{k_1, k_2\} \geq 2$ such that $M^+_{k_1} \cup M^-_{k_2} \subseteq C$.

Combining Theorem 3.5 and Lemma 3.6 we get a full characterization for the computational complexity of UPQR Necessary Manipulation with conclusions chosen from a standard-form clause set C.

According to the definition, an instance of UPQR Necessary Manipulation with conclusions chosen from C is a yes-instance if and only if there is one target conclusion $C^* \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ such that C^* and all formulas in $Q_0 = \mathrm{UPQR}_q(\boldsymbol{J}) \cap J$ can be included in the manipulated outcome at the same time. Note that we already know that all formulas in Q_0 are in the original outcome $\mathrm{UPQR}_q(\boldsymbol{J})$, which means that all formulas in Q_0 can be satisfied at the the same time. The question is whether it is possible to satisfy one more clause $C^* \notin Q_0$. Therefore, Theorem 3.5 implies that this additional information does not help to efficiently determine whether all conclusions in $Q = Q_0 \cup \{C^*\}$ can be satisfied at the same time. This is the main idea for the proofs of the following Lemmas 3.7 and 3.9.

According to Lemma 3.6, we need to consider two cases. We first prove a weaker version of Theorem 3.5 in the following.

Lemma 3.7. *If $(M^+_{k_1} \cup M^-_{k_2})$-Sat is NP-complete, then UPQR Necessary Manipulation with conclusions chosen from a closely related standard-form clause set $C = M^+_{k_1+1} \cup M^-_{k_2}$ is NP-complete.*

Proof. We first show the result for $q = \frac{1}{2}$. We present a polynomial-time reduction from $(M^+_{k_1} \cup M^-_{k_2})$-Sat to UPQR Necessary Manipulation with conclusions chosen from C. Given an instance

$$C_1^+ \wedge \ldots \wedge C^+_{m_1} \wedge C_1^- \wedge \ldots \wedge C^-_{m_2}$$

of $(M^+_{k_1} \cup M^-_{k_2})$-Sat, where $C_i^+ \in M^+_{k_1}$ and $C_i^- \in M^-_{k_2}$, we construct an instance of Manipulation as in Table 1. The agenda contains all variables $x_1, \ldots, x_n$ that appear in the $(M^+_{k_1} \cup M^-_{k_2})$-Sat instance and their negations. In addition, we create $y_1, \ldots, y_{k_2}$ and their negations in premises. Then we add $C_i^+ \vee y_1$ for $1 \leq i \leq m_1$, C_i^- for $1 \leq i \leq m_2$, $\neg y_1 \vee \ldots \vee \neg y_{k_2}$ and their negations as conclusions. The set of judges is $N = 1, 2, 3$. The manipulator is the third judge and his desired set J consists of all positive conclusions. The manipulator is decisive for variables $x_1, \ldots, x_n$ and y_1.

Since all positive conclusions except for $\neg y_1 \vee \ldots \vee \neg y_{k_2}$ are already in the truthful outcome $\mathrm{UPQR}_{1/2}(\boldsymbol{J})$, to make a successful manipulation, the manipulator has to make the manipulated outcome contain all positive conclusions. Specifically, for conclusion $\neg y_1 \vee \ldots \vee \neg y_{k_2}$, since $y_j = 1, j \geq 2$ can not be changed by the manipulator, the manipulator has to set $y_1 = 0$. Then, to satisfy all remaining conclusions $C_i^+ \vee y_1$ and C_i^- is equivalent to setting values for $x_1, \ldots, x_n$ to satisfy $C_1^+ \wedge \ldots \wedge C^+_{m_1} \wedge C_1^- \wedge \ldots \wedge C^-_{m_2}$.

For other rational quota q and any fixed number $m \geq 3$ of judges, this proof still works with minor modifications as follows. The agenda remains the same, and the judgment set of the manipulator is equal to J_3. For other judges, the first $\lfloor mq \rfloor$ judgment sets are equal to J_1, and the remaining judgment sets are equal to J_2. □

As a corollary, we get that the respective Manipulation problems "corresponding to" 3-Sat and Monotone-Sat are NP-complete.

Corollary 3.8. *UPQR Necessary Manipulation with conclusions chosen from $\cup_j S_3^j$ or from $\cup_{k=1}^{\infty}(M_k^+ \cup M_k^-)$ is NP-complete.*

Note that in Lemma 3.7 the two clause sets $M^+_{k_1} \cup M^-_{k_2}$ (in Satisfiability) and $M^+_{k_1+1} \cup M^-_{k_2}$ (in Manipulation) are not the same. This leaves a gap when conclusions of UPQR Necessary Manipulation are chosen from $M_2^+ \cup M_3^-$ (or equivalently $M_3^+ \cup M_2^-$): We cannot adopt Lemma 3.7, since the corresponding Satisfiability problem is $(M_1^+ \cup M_3^-)$-SAT (or $(M_2^+ \cup M_2^-)$-SAT) which is not NP-complete (cf. Lemma 3.6). Next we close this gap by giving a more

involved reduction to show the NP-hardness for the case when conclusions are chosen from $M_2^+ \cup M_3^-$.

Lemma 3.9. *UPQR Necessary Manipulation with conclusions chosen from $(M_2^+ \cup M_3^-)$ is NP-complete.*

Proof. We show the result for $q = \frac{1}{2}$. For other values of q the result can be shown by the same reduction with minor modifications. We present a polynomial-time reduction from $(M_2^+ \cup M_3^-)$-SAT.

Given an instance $f_1 \wedge f_2$ of $(M_2^+ \cup M_3^-)$-SAT, where f_1 is a conjunction of clauses of the form "$x_{i_1} \vee x_{i_2}$" from M_2^+ with $x_{i_1}, x_{i_2} \in \{x_1, \ldots, x_n\}$ and f_2 is a conjunction of clauses of the form "$\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$" from M_3^- with $x_{i_1}, x_{i_2}, x_{i_3} \in \{x_1, \ldots, x_n\}$, we construct an instance of UPQR Necessary Manipulation with conclusions chosen from $M_2^+ \cup M_3^-$ as follows (see also Table 2).

- For every original variable $x_i, 1 \le i \le n$ create the premises x_i, y_i, z_i (and their negations).
- Create two premises w and v (and their negations), and create the clause $w \vee v$ (and its negation) in the conclusions.
- For every original clause $x_{i_1} \vee x_{i_2}$ in f_1, create the clause $z_{i_1} \vee z_{i_2}$ (and its negation) in the conclusions. Note that original variables $x_i, 1 \le i \le n$ are replaced by premises $z_i, 1 \le i \le n$.
- For every original clause $\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$ in f_2, create the clause $\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$ (and its negation) in the conclusions.
- For each i with $1 \le i \le n$, create four clauses $x_i \vee y_i$, $y_i \vee z_i$, $\neg x_i \vee \neg y_i \vee \neg w$ and $\neg y_i \vee \neg z_i \vee \neg w$ (and their negations) in the conclusions.

The set of judges is $N = 1, 2, 3$. The manipulator is the third judge and his desired set J consists of all positive conclusions. The manipulator is decisive for all variables except for v. We now show that $f_1 \wedge f_2$ is satisfiable if and only if the manipulation is feasible.

$\Rightarrow$ Assume that $f_1 \wedge f_2$ is satisfiable, then there is a value assignment $x_i^*, 1 \le i \le n$ such that all clauses in $f_1 \wedge f_2$ are satisfied. So the manipulator can set $x_i = z_i = x_i^*, 1 \le i \le n$ to satisfy conclusions $\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$ and $z_{i_1} \vee z_{i_2}$. All remaining positive conclusions can be satisfied by setting $w = 0$ and $y_i = -x_i^*$ for $1 \le i \le n$. Thus the manipulation is feasible. Recall that the manipulator is decisive for all variables except for v.

$\Leftarrow$ Assume that the manipulation is feasible. Since all positive conclusions, except for $w \vee v$, are already in the truthful outcome $\mathrm{UPQR}_{1/2}(\boldsymbol{J})$, the manipulation is feasible means that there is a value assignment for all variables with $v = 0$ (since v is not by the manipulator) such that all positive conclusions can be satisfied. Specifically, for conclusion $w \vee v$, since $v = 0$ can not be changed by the manipulator, the manipulator has to set $w = 1$. Then $\neg x_i \vee \neg y_i \vee \neg w$ and $\neg y_i \vee \neg z_i \vee \neg w$ are equivalent to $\neg x_i \vee \neg y_i$ and $\neg y_i \vee \neg z_i$, respectively. Together with $x_i \vee y_i$ and $y_i \vee z_i$, we have

$$(\neg x_i \vee \neg y_i) \wedge (x_i \vee y_i) \Rightarrow x_i = -y_i,$$
$$(\neg y_i \vee \neg z_i) \wedge (y_i \vee z_i) \Rightarrow y_i = -z_i.$$

This means in this value assignment $x_i = z_i$. Since all conclusions $\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$ and $z_{i_1} \vee z_{i_2}$ can be satisfied by this value assignment with $x_i = z_i$, we have that $f_1 \wedge f_2$ is satisfiable. □

Due to the limited space, we refer to the full version of this paper for the remaining part of the proof (the first case in Lemma 3.6) of Theorem 3.5. In the above two reductions in Lemmas 3.7 and 3.9, to make a successful manipulation, the manipulator has to make the manipulated outcome contain all positive conclusions. Thus these reductions also work for UPQR Exact Manipulation if we choose the desired set to consist of all positive conclusions.

Proposition 3.10 (★). *For any standard-form clause set C, UPQR Exact Manipulation with conclusions chosen from C and C-Sat are equivalent under polynomial-time Turing reductions.*

4 HAMMING DISTANCE BASED MANIPULATION

We now move on to UPQR HD Manipulation which is the very first variant of Manipulation analyzed by Endriss et al. [12] for the majority threshold $q = 1/2$. In UPQR HD Manipulation, the manipulator cares about the number of formulas in the desired set J achieved by the collective judgment set. The formal definition is given as follows.

UPQR HD Manipulation

Input: An agenda Φ, a profile $\boldsymbol{J} = (J_1, \ldots, J_n) \in \mathcal{J}(\Phi)^n$, the manipulator's desired consistent (possibly incomplete) set $J \subseteq J_n$, and a uniform rational threshold $q \in [0, 1)$.

Question: Does there exist a judgment set $J^* \in \mathcal{J}(\Phi)$ such that $\mathrm{HD}(J, \mathrm{UPQR}_q(\boldsymbol{J}_{-n}, J^*)) < \mathrm{HD}(J, \mathrm{UPQR}_q(\boldsymbol{J}))$?

Herein, the Hamming distance $\mathrm{HD}(J, S)$ between the possibly incomplete desired set J and a complete collective judgment set S is the number of formulas in J which are not contained in S, i.e. $\mathrm{HD}(J, S) = |J \setminus S|$.

Without loss of generality, in this section we assume that $J = J_n \cap \Phi_c$, that is, the desired set contains all conclusions from J_n but no premise: Every instance of UPQR HD Manipulation can be easily transformed into an equivalent instance with $J = J_n \cap \Phi_c$ as follows. If for some conclusion φ none of φ and $\neg\varphi$ appears in J, then just delete φ and $\neg\varphi$ from the agenda. If there is some premise x with $x \in J$ (or $\neg x \in J$), we can remove it from J, create two clauses $x \vee x'$ and $\neg(x \vee x')$ in the conclusions and adding $x \vee x'$ (or $\neg(x \vee x')$) to J, where x' is a dummy variable with $x' \notin J_i$ for all $1 \le i \le n$. Note that doing so we just add positive monotone clauses with two literals $(x \vee x')$ into the conclusion set.

Baumeister et al. [4] proved that UPQR HD Manipulation is NP-complete for positive monotone clauses. In this section we show that this problem is NP-complete even for positive monotone clauses of length $\ell = 3$ by reducing from a natural variant of Vertex Cover which could be interesting on its own. When the clause length is 2, we show the problem is in P for positive monotone clauses, but NP-complete for monotone clauses or Horn clauses.

4.1 Condition for a successful manipulation

In this section we give a sufficient and necessary condition for a successful manipulation in Lemma 4.2. We first classify all variables into the following four different classes with respect to different combinations of its value in the truthful outcome $\mathrm{UPQR}_q(\boldsymbol{J})$ and the judgment set of the manipulator J_n:

(1) $P_1^1 = \{x \in \Phi_p \mid x \in J_n \wedge x \in \mathrm{UPQR}_q(\boldsymbol{J})\}$;
(2) $P_1^0 = \{x \in \Phi_p \mid x \notin J_n \wedge x \in \mathrm{UPQR}_q(\boldsymbol{J})\}$;
(3) $P_0^0 = \{x \in \Phi_p \mid x \notin J_n \wedge x \notin \mathrm{UPQR}_q(\boldsymbol{J})\}$;
(4) $P_0^1 = \{x \in \Phi_p \mid x \in J_n \wedge x \notin \mathrm{UPQR}_q(\boldsymbol{J})\}$.

Table 2: Instance of UPQR Necessary Manipulation with conclusion set $C = M_2^+ \cup M_3^-$ for the proof of Lemma 3.9.

Judgment Set	x_i	y_i	z_i	w	v		$w \vee v$	$\neg x_{i_1} \vee \neg x_{i_2} \vee \neg x_{i_3}$	$\neg x_i \vee \neg y_i \vee \neg w$	$\neg y_i \vee \neg z_i \vee \neg w$	$x_i \vee y_i$	$y_i \vee z_i$	$z_{i_1} \vee z_{i_2}$
J_1	1	1	1	1	0		1	0	0	0	1	1	1
J_2	0	0	0	0	0		0	1	1	1	0	0	0
J_3	0	1	1	0	1		1	1	1	1	1	1	1
$UPQR_{1/2}$	0	1	1	0	0	$\Rightarrow$	0	1	1	1	1	1	1

Judgment Set	$x \in P_1^1$	$x \in P_1^0$	$x \in P_0^0$	$x \in P_0^1$
$\boldsymbol{J}_{-n}$	*	*	*	*
J_n	1	0	0	1
$\mathrm{UPQR}_q(\boldsymbol{J})$	1	1	0	0

Observation 1. *Variables from $P_0^1 \cup P_1^0$ are not decided by the manipulator.*

In the following, a variable is called *useful* if it is decided by the manipulator and changing its value can make the outcome contain at least one new conclusion from $J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$, where $J = J_n \cap \Phi_c$ is the manipulator's desired set. For a variable $x \in P_1^1$, any positive monotone clause φ containing x is already in $J \cap \mathrm{UPQR}_q(\boldsymbol{J})$, thus changing the value of x from 1 to 0 cannot make the outcome contain any new conclusion from $J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$. Therefore, all useful variables are from P_0^0.

Observation 2. *If x is a useful variable, then $x \in P_0^0$.*

Definition 4.1. A positive conclusion φ is called *good* if $\varphi \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ and is called *bad* if $\neg\varphi \in J \cap \mathrm{UPQR}_q(\boldsymbol{J})$.

Note that a good (or bad) conclusion is a candidate for decreasing (or increasing) the Hamming distance $\mathrm{HD}(J, \mathrm{UPQR}_q(\boldsymbol{J}))$.

Example. Consider the following profile:

Judg. Set	x_1	x_2	x_3	x_3'	x_4		$x_1 \vee x_2$	$x_2 \vee x_3$	$x_3 \vee x_3'$	$x_3 \vee x_4$
J_1	1	0	1	1	1		1	1	1	0
J_2	0	0	0	0	1		0	0	0	1
J_3	1	1	0	0	0		1	1	0	0
$UPQR_{1/2}$	1	0	0	0	1	$\Rightarrow$	1	0	0	1

Variables x_1, x_3, and x_3' are decided by the manipulator, but $x_1 \in P_1^1$ is not useful since change its value from 1 to 0 would only exclude $x_1 \vee x_2 \in J$ from the outcome. Conclusion $x_2 \vee x_3$ is good since it is in $J_3 \setminus \mathrm{UPQR}_q(\boldsymbol{J})$, and changing x_3 from 0 to 1 will make x_3 and $x_2 \vee x_3$ included in the outcome. Conclusion $x_3 \vee x_3'$ is bad since its negation $\neg(x_3 \vee x_3') \in J_3 \setminus \mathrm{UPQR}_q(\boldsymbol{J})$, and changing x_3 or x_3' from 0 to 1 will make $x_3 \vee x_3'$ included in the outcome, and hence $\neg(x_3 \vee x_3')$ is excluded from the outcome.

Now we give a sufficient and necessary condition for a successful manipulation.

Lemma 4.2. *An instance of UPQR HD Manipulation with all conclusions being positive monotone clauses is a yes-instance if and only if there is a set $S \subseteq P_0^0$ of useful variables, such that after changing their values from 0 to 1, the number of good conclusions included in the outcome is strictly larger than the number of bad conclusions included in the outcome:*

$$|\{\varphi \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J}) \mid S_\varphi \cap S \neq \emptyset\}| > |\{\neg\varphi \in J \cap \mathrm{UPQR}_q(\boldsymbol{J}) \mid S_{\neg\varphi} \cap S \neq \emptyset\}|,$$

where S_φ ($S_{\neg\varphi}$) is the set of variables appearing in clause φ ($\neg\varphi$).

Proof. According to the definition of useful variables, if an instance of UPQR HD Manipulation is a yes-instance, then the manipulator can achieve a better outcome by changing only the values of useful variables. According to Observation 2, all useful variables are from P_0^0. To prove this lemma, it suffices to show why we just need to consider good conclusions and bad conclusions. If a positive conclusion φ is neither good nor bad, then it must be $\neg\varphi \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ or $\varphi \in J \cap \mathrm{UPQR}_q(\boldsymbol{J})$. In both cases we have that $\varphi \in \mathrm{UPQR}_q(\boldsymbol{J})$. Changing the values of variables in P_0^0 from 0 to 1 will not change the value of φ (φ is still in $\mathrm{UPQR}_q(\boldsymbol{J})$ after this change). Therefore, we just need to consider the influence on the number of good conclusions and bad conclusions after changing the values of useful variables. □

4.2 Positive monotone clauses of length $\ell = 2$

In this section we show UPQR HD Manipulation with positive monotone clauses of length $\ell = 2$ is solvable in polynomial time by a reduction to the Weighted Maximum Density Subgraph (WMDS) problem. Given an undirected graph $G = (V, E)$ with nonnegative rational edge weights $w(e)$ and vertex weights $w(v)$, and a nonnegative rational number k, WMDS asks to decide the existence of a vertex subset $V' \subseteq V$ with $\sum_{v \in V'} w(v) > 0$ such that

$$\frac{\sum_{e \in E(G[V'])} w(e)}{\sum_{v \in V'} w(v)} > k,$$

where $G[V']$ is the subgraph induced by V'. Goldberg [14] shows that WMDS can be solved in polynomial time by a reduction to the Minimum Cut problem.

Theorem 4.3. *UPQR HD Manipulation with positive monotone clauses of length $\ell = 2$ is solvable in polynomial time.*

Proof. According to Lemma 4.2, we need to find a set of useful variables in P_0^0 such that after changing the value of these variables the number of good conclusions included in the outcome is strictly larger than the number of bad conclusions included in the outcome. Every good conclusion $\varphi \in J \setminus \mathrm{UPQR}_q(\boldsymbol{J})$ contains at least one variable x such that $x \in J_n$ since $\varphi \in J \subseteq J_n$. Moreover, since $\varphi \notin \mathrm{UPQR}_q(\boldsymbol{J})$ we have $x \notin \mathrm{UPQR}_q(\boldsymbol{J})$. Thus φ contains at least one variable $x \in P_0^1$, which is not decided by the manipulator according to Observation 1. Hence a good conclusion φ of length 2 contains at most one variable which is possibly decided by the manipulator. However, a bad conclusion of length 2 may contain two useful

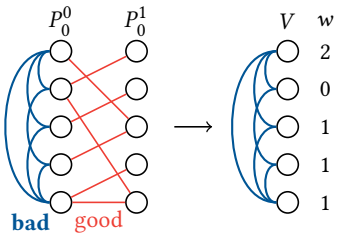

Figure 1: Illustration of the constructed weighted graph in the proof of Theorem 4.3. On the left side a vertex represents a variable from P_0^0 or P_0^1. A line between two vertices represents a (good or bad) conclusion containing the two corresponding variables. We transform it into the vertex weighted graph on the right side, where the vertex set V corresponds to P_0^0 and the weight w for a vertex $v \in V$ is the difference between the number of bad and good conclusions that contain the corresponding variable x_v.

variables from P_0^0. Thus, if we change the values of a set of variables from P_0^0 and sum up the number of included bad conclusions, then some bad conclusions will be counted twice. To solve this issue, we create a weighted graph $G = (V, E)$ as follows (see also Figure 1): First, for every useful variable $x \in P_0^0$, create a vertex $v \in V$ and assign it a weight $w(v) = n_v - p_v$, where n_v is the number of bad conclusions containing x and p_v is the number of good conclusions containing x. Thus $w(v)$ is the increased Hamming distance when a single variable x is changed. Second, for every pair of vertices u and v, create an edge between them if there is a bad conclusion $\varphi = x_u \vee x_v$, where x_u and x_v are the corresponding variables of u and v.

We first do the following preprocessing. If there is a vertex $v \in V$ with $w(v) < 0$, then changing this variable alone can strictly decrease the Hamming distance and hence the manipulation is feasible. If there is an edge $e = \{u, v\}$ with $w(u) = w(v) = 0$, then changing the value of x_u and x_v can decrease the Hamming distance by 1 and hence the manipulation is feasible. So in the following we can assume $w(v) \geq 0$ for every $v \in V$ and there is no edge $e = \{u, v\}$ with $w(u) = w(v) = 0$. For any vertex subset $V' \subseteq V$, changing the value of the corresponding variables can increase the distance by $\sum_{v \in V'} w(v) - |E(G[V'])|$, where $G[V']$ is the subgraph induced by V'. If $\sum_{v \in V'} w(v) = 0$, then according to the above assumption, we have $|E(G[V'])| = 0$. Therefore manipulation is feasible if and only if there is a vertex subset V' with $\sum_{v \in V'} w(v) > 0$ such that $\sum_{v \in V'} w(v) - |E(G[V'])| < 0$ or $|E(G[V'])| / \sum_{v \in V'} w(v) > 1$. This is just an instance of the WMDS problem with edge weight 1, which can be solved in polynomial time [14]. □

4.3 Positive monotone clauses of length $\ell = 3$

In this section we show that UPQR HD Manipulation with positive monotone clauses of length $l = 3$ is NP-complete. The main difference between $l = 2$ and $l = 3$ is that when $l = 2$, every good conclusion must contain a variable from P_0^1, and hence contains at most one useful variable from P_0^0. When $l \geq 3$, however, in addition to one variable from P_0^1, a good conclusion can contain two useful variables from P_0^0. Hence, useful variables are not independent with respect to good conclusions. See also Figure 2 for the comparison.

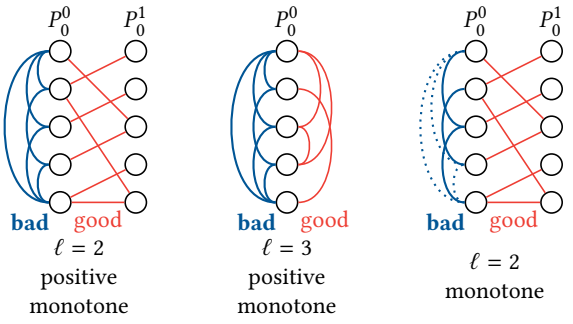

Figure 2: Comparison between different clause classes. A vertex represents a variable from P_0^0 or P_0^1, and only variables from P_0^0 could be decided by the manipulator. A line between two vertices represents a conclusion containing the two corresponding variables. A line is solid if changing the value of *one* of its endpoints in P_0^0 will change the value of this conclusion, while a line is dotted if changing the value of *both* endpoints in P_0^0 will change the value of this conclusion.

Figure 2 shows that when $\ell = 3$, then we need to find a vertex subset of P_0^0 to cover more good (red) edges than bad (bold blue) edges. This leads us to the following closely related graph problem.

Positive Vertex Cover
Input: An undirected graph $G = (V, E^+ \cup E^-)$ with $E^+ \cap E^- = \emptyset$.
Question: Is there a vertex subset $V' \subseteq V$ which covers strictly more edges in E^+ than in E^-?

Lemma 4.4 (★). *Positive Vertex Cover is NP-complete.*

Proof Sketch. We construct a reduction from Cubic Vertex Cover, where given an undirected 3-regular graph and an integer k, the task is to determine whether there exists a vertex cover of size at most k. Given an instance $(G_0 = (V_0, E_0), k)$ of Cubic Vertex Cover. Denote $n = |V_0|$. Since a vertex cover needs at least $\frac{n}{2}$ vertices, we can assume $\frac{n}{2} \leq k \leq n$. We create an instance $G = (V, E^+ \cup E^-)$ of Positive Vertex Cover as follows (see also Figure 3). First, for every original vertex v_i in V_0, create a vertex x_i in V, and for every edge $\{v_i, v_j\}$ in E_0, create an edge $\{x_i, x_j\}$ in E^+. Then, create three more vertices y_1, y_2, y_3 in V and create edges $\{x_i, y_1\}$, $\{x_i, y_2\}$ and $\{x_i, y_3\}$ for every $1 \leq i \leq n$ in E^-. Add an edge $\{y_1, y_2\}$ in E^+. Next, create vertices $z_1, \ldots, z_{n-p}$ in V, where $p = \frac{3n}{4} - \frac{k}{2}$, and create edges $\{z_j, y_1\}$ and $\{z_j, y_2\}$ for every $1 \leq j \leq n - p$ in E^+. Finally, create vertices w_1, w_2, w_3 in V, and create edges $\{z_j, w_1\}$, $\{z_j, w_2\}$ and $\{z_j, w_3\}$ for every $1 \leq j \leq n - p$ in E^-.

It is easy to see that w_1, w_2, w_3, y_3 and z_j with $1 \leq j \leq n - p$ will never be chosen, so our choice is constrained in $\{x_1, \ldots, x_n, y_1, y_2\}$. We can show that: If the number of vertices chosen from $\{x_1, \ldots, x_n\}$ is less than p, then it is always better to not choose y_1 or y_2. Otherwise, it is always better to choose y_1 and y_2. Now one can verify this reduction as follows. If there is a vertex cover of size $k^* \leq k$, then the corresponding k^* vertices in V together with y_1 and y_2 cover more edges in E^+ than in E^-. Conversely, if there is a vertex subset V^* that covers more edges in E^+ than in E^-, then V^* has to contain at least p vertices, which means $y_1, y_2 \in V^*$. Then we can argue that the remaining vertices in V^* from $\{x_1, \ldots, x_n\}$ must

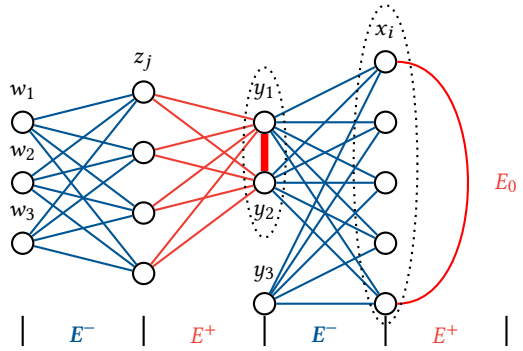

Figure 3: Illustration of the constructed instance in the proof of Lemma 4.4. Bold blue edges are edges in E^- and red edges are edges in E^+.

Table 3: Example for Monotone clause

Judg. Set	x_1	x_2	$\neg x_1 \vee \neg x_2$
J_1	1	1	0
J_2	0	0	1
J_3	0	0	1
$UPQR_{1/2}$	0	0	$\Rightarrow 1$

Table 4: Example for Horn clause

Judg. Set	x_1	x_2	$\neg x_1 \vee x_2$
J_1	1	0	0
J_2	0	0	1
J_3	1	1	1
$UPQR_{1/2}$	1	0	$\Rightarrow 0$

cover enough edges in E_0 such that we can easily extend it to a vertex cover for G_0 of size at most k. □

Now we can show the NP-hardness of UPQR HD Manipulation with positive monotone clauses of length $\ell = 3$ by a simple reduction from Positive Vertex Cover.

Theorem 4.5 (★). *UPQR HD Manipulation with positive monotone clauses of fixed length l (≥ 3) is NP-complete.*

4.4 Monotone or Horn clauses of length $\ell = 2$

When clauses are not positive monotone, we can not use the characterization for a successful manipulation given in Lemma 4.2.

For monotone clauses we may have both $x_i \vee x_j$ and $\neg x_i \vee \neg x_j$ in the conclusions. As shown in the example in Table 3, conclusion $\neg x_1 \vee \neg x_2$ will be excluded from the outcome only when both x_1 and x_2 have been changed. Recall that for positive monotone clauses, changing one variable is enough to include a bad conclusion (see also Figure 2 for the comparison). So for monotone clauses we have a new kind of "bad" conclusions.

For Horn clauses, we have conclusions of the form $\neg x_i \vee x_j$. This allows variables from P_1^1 to be useful. To see this, consider the example in Table 4 where J_3 is the manipulator and $J = \{\neg x_1 \vee x_2\}$ is the desired set. Changing the value of $x_1 \in P_1^1$ from 1 to 0 can make $\neg x_1 \vee x_2$ included in the outcome. So for Horn clauses we have a new kind of useful variables.

With these differences, we can show the following theorem.

Theorem 4.6 (★). *UPQR HD Manipulation with monotone clauses or Horn clauses of length $l = 2$ is NP-complete.*

Table 5: Computational complexity of basic variants of Manipulation.

UPQR-M-Manipulation $M =$	Possible / Robustness	Necessary / Exact
no restriction	NP-c [4]	NP-c [4]
standard-form clause set C	P (Lem. 3.1)	C-Sat (Thm. 3.5 Pro. 3.10)
monotone clauses	P (Cor. 3.2)	NP-c (Cor. 3.8)
clauses with length $\ell \leq 3$	P (Cor. 3.2)	NP-c (Cor. 3.8)
Horn clauses	P (Cor. 3.2)	P (Cor. 3.4)
positive monotone clauses	P (Cor. 3.2)	P (Cor. 3.4)

5 CONCLUSION

This paper provides a refined picture in terms of the computational complexity of different variants of Manipulation in judgment aggregation. Our results for basic variants of Manipulation are summarized in Table 5. [3]

For UPQR HD Manipulation, we show that NP-hardness holds even if all conclusions are positive monotone clauses with length $\ell = 3$ but that the problem becomes solvable in polynomial time when $\ell = 2$. For monotone or Horn clauses with $\ell = 2$, the problem is also NP-hard which is in stark contrast to all basic variants of Manipulation that remain polynomial-time solvable for Horn and positive monotone clauses of arbitrary length.

All Manipulation variants we considered were known to be generally NP-hard, which was seen and sold as "barrier against manipulative behavior" [4]. The main message of this work is that several basic variants of Manipulation can be solved efficiently for simple but well-motivated restrictions of conclusions (e.g. Horn clauses and generalizations thereof) whereas other variants remain computationally intractable for most restrictions. We see our results as an important step and expect further effects decreasing the computational complexity by considering other realistic structural properties of the formulas (e.g. bounded frequency of variables). Hence, our results question whether there really is a barrier against manipulative behavior in case of realistically simple formulas.

Possible next steps include a systematic investigation of the parameterized complexity for both judgment aggregation-specific parameters (e.g. "number of judges" or "size of the desired set") and formula specific parameters (e.g. "number of clauses" or "variable frequency"). We note that considering the parameter "number of judges" alone, however, will not lead to tractable cases because this parameter is fixed to three in many of our reductions. Furthermore, it seems natural to extend the study to strategic behavior of groups of judges instead of a single judge [7].

ACKNOWLEDGMENTS

Junjie Luo was supported by CAS-DAAD Joint Fellowship Program for Doctoral Students of UCAS. Work done while all authors were with TU Berlin.

[3] We remark that our results provide also a full picture for the omitted variants of Manipulation as defined by Baumeister et al. [4]. For UPQR-TR-Possible-Manipulation we can obtain the same results as for UPQR Possible Manipulation. UPQR-U-Necessary-Manipulation is known to be trivial to solve [4].

REFERENCES

1. John J. Bartholdi III, Craig A. Tovey, and Michael A. Trick. 1989. The Computational Difficulty of Manipulating an Election. *Social Choice and Welfare* 6, 3 (1989), 227–241.
2. John J. Bartholdi, III, Craig A. Tovey, and Michael A. Trick. 1992. How Hard Is It to Control an Election? *Mathematical and Computer Modeling* 16, 8-9 (1992), 27–40.
3. Dorothea Baumeister, Gábor Erdélyi, Olivia J. Erdélyi, and Jörg Rothe. 2013. Computational Aspects of Manipulation and Control in Judgment Aggregation. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT '13) (LNCS)*, Vol. 8176. Springer, 71–85.
4. Dorothea Baumeister, Gábor Erdélyi, Olivia J. Erdélyi, and Jörg Rothe. 2015. Complexity of manipulation and bribery in judgment aggregation for uniform premise-based quota rules. *Mathematical Social Sciences* 76 (2015), 19–30.
5. Dorothea Baumeister, Gábor Erdélyi, and Jörg Rothe. 2016. Judgment Aggregation. In *Economics and Computation*. Chapter 8, 361–391.
6. Dorothea Baumeister, Jörg Rothe, and Ann-Kathrin Selker. 2017. Strategic Behavior in Judgment Aggregation. In *Trends in Computational Social Choice*, Ulle Endriss (Ed.). AI Access, Chapter 8, 145–168.
7. Sirin Botan, Arianna Novaro, and Ulle Endriss. 2016. Group Manipulation in Judgment Aggregation. In *Proceedings of the 15th International Conference on Autonomous Agents & Multiagent Systems (AAMAS '16)*. ACM, 411–419.
8. Stefano Ceri, Georg Gottlob, and Letizia Tanca. 2012. *Logic programming and databases*. Springer Science & Business Media.
9. Franz Dietrich and Christian List. 2007. Judgment aggregation by quota rules: Majority voting generalized. *Journal of Theoretical Politics* 19, 4 (2007), 391–424.
10. Franz Dietrich and Christian List. 2007. Strategy-proof judgment aggregation. *Economics and Philosophy* 23, 3 (2007), 269–300.
11. Ulle Endriss. 2016. Judgment aggregation. In *Handbook of Computational Social Choice*, Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia (Eds.). Cambridge University Press, Chapter 17.
12. Ulle Endriss, Umberto Grandi, and Daniele Porello. 2012. Complexity of judgment aggregation. *Journal of Artificial Intelligence Research* 45 (2012), 481–514.
13. Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. 2011. The Shield That Never Was: Societies With Single-Peaked Preferences Are More Open to Manipulation and Control. *Information and Computation* 209, 2 (2011), 89–107.
14. Andrew V. Goldberg. 1984. *Finding a maximum density subgraph*. University of California Berkeley, CA.
15. Davide Grossi and Gabriella Pigozzi. 2014. *Judgment Aggregation: A Primer*. Morgan & Claypool Publishers.
16. Lewis A. Kornhauser and Lawrence G. Sager. 1986. Unpacking the Court. *The Yale Law Journal* 96, 1 (1986), 82–117.
17. Christian List. 2012. The theory of judgment aggregation: an introductory review. *Synthese* 187, 1 (2012), 179–207.
18. Christian List and Clemens Puppe. 2009. Judgment Aggregation: A Survey. In *Handbook of Rational and Social Choice*, Christian List and Clemens Puppe (Eds.). Oxford University Press, Chapter 19.
19. John W. Lloyd. 1987. *Foundations of Logic Programming*. Springer-Verlag.